Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Blaž Dolenc

Geografska segmentacija uporabnikov za uporabo v oglaševanju

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Blaž Zupan

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomski nalogi na osnovi podatkov poiščite geografsko smiselne skupine spletnih uporabnikov, za katere so podani podatki o klikih na oglase, lokacije uporabnikov in dodatni demografski parametri, ki opisujejo posamezne lokacije. Skupine potem uporabite v namene napovedovanja, ali bo uporabnik kliknil na izbrani oglas. Metodološke rešitve uporabite in preverite na izzivu podjetja Zemanta s področja geografske segmentacije uporabnikov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Blaž Dolenc sem avtor diplomskega dela z naslovom:

Geografska segmentacija uporabnikov za uporabo v oglaševanju

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Blaža Zupana.
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 10. septembra 2015 Podpis avtorja:

Zahvaljujem se mentorju prof. dr. Blažu Zupanu za strokovno pomoč in ideje pri izdelavi diplomske naloge. Zahvala gre tudi podjetju Zemanta za dovoljenje za uporabo podatkov pri diplomskem delu. Posebej se zahvaljujem družini, prijateljem in dekletu za podporo med študijem in pisanjem diplomske naloge.

Kazalo

Povzetek

Abstract

1 Uvod **1**

1.1 Cilji 2

2 Opis problema **3**

3 Odkrivanje skupin **5**

3.1 Pregled področja in tehnik iskanja skupin 5

3.2 Podatki 6

3.3 Uporabljene metode in izvedba 8

3.4 Prikaz in interpretacija skupin 14

3.5 Rezultati iskanja skupin 18

4 Napovedovanje **23**

4.1 Pregled področja in metod napovedovanja 24

4.2 Podatki 27

4.3 Uporabljene metode in praktična izvedba 28

4.4 Rezultati 33

5 Sklepne ugotovitve **39**

Povzetek

V današnjem spletnem oglaševanju ni več edini cilj prikazati oglasa čim večjemu številu potencialnih kupcev, temveč si oglaševalci vse bolj prizadevajo oglas prikazati tistemu, ki ga bo najverjetneje zanimal. Na primer, če poznamo uporabnikovo okvirno lokacijo, lahko na podlagi prejšnjih obiskovalcev napovemo klik oglasa. Potrebo po geografski segmentaciji uporabnikov so zaznali tudi pri podjetju Zemanta, kjer so študentom zastavili izziv, pri katerem je bilo potrebno obiskovalce spletnih strani razdeliti glede na poštno številko iz katere prihajajo, ter to uporabiti kot podlago za napoved klika. Cilj naloge je bilo poiskati čim bolj smiselne skupine uporabnikov, ter jih ustrezno predstaviti, v drugem delu pa zgraditi napovedni model za napovedovanje klika na oglas, ki bo dosegal točnost napovedi AUC okoli 0,75. V nalogi poročamo o naši rešitvi tega problema, ki uporablja vrsto tehnik s področja strojnega učenja. Končna razdelitev uporabnikov, ki jo predlagamo, je obsegala 20 skupin, ki so se med seboj močno razlikovale glede na gostoto poselitve, urbanizacije in ostalih demografskih dejavnikov. Prikaz skupin na zemljevidu je pokazal, da je razdelitev smiselna. Končni AUC na testnih podatkih je znašal 0,79.

Ključne besede: Iskanje skupin v podatkih, gručenje, strojno učenje.

Abstract

In modern web advertising the goal is not only deliver an ad to a broad number of customers, but to target particular customers who are more likely to be interested in content. If the user location is known, we can estimate click on ad based on previous visitors. The company Zemanta recognized the need for geographic audience segmentation, and they have invited students to solve their challenge. The goal was geographic segmentation of web pages visitors based on the ZIP code they come from and development of a prediction model, which can estimate the probability of click on the ad, with accuracy (AUC score) around 0,75. In this dissertation, we describe our the solution to the challenge. Our user segmentation identified 20 groups. There were large differences between them considering population density, urbanization and other demographic indicators. Plotting results on map revealed, that segmentation is meaningful. Our final AUC score on test data was 0,79.

Keywords: Clustering, Data mining, Machine learning.

Poglavje 1

Uvod

Področje strojnega učenja in odkrivanja znanja v podatkih oziroma podatkovnega rudarjenja se v zadnjih letih bliskovito razvija [2]. Na to vpliva več dejavnikov, najpomembnejši pa je potreba podjetij po boljšem poznavanju svojih kupcev in doseganju konkurenčnih prednosti, ki jih uporaba teh tehnik omogoča [12].

Obdelava in uporaba podatkov je še posebej vse prisotna na spletu. Večini spletnih strani in družabnih omrežij je (vsaj delni) vir prihodka prikaz oglasov, poleg prikaza pa je zelo pomemben tudi odziv uporabnika, torej klik na oglas. Oglaševalci si želijo optimizirati stroške prikaza oglasa, in ga ponuditi le tistim, za katerega verjamejo, da jih oglas utegne zanimati. Posledično so se razvile številne oglaševalske platforme[1], katerih cilj je prav to - optimizirati prikaz oglasov in vsebin pravim uporabnikom.

Osnova za diplomsko delo je programerski izziv podjetja Zemanta[2], ki se ukvarja z razvojem platforme za dostavo oglasnih vsebin. Pri tem se močno zanašajo na strojno učenje in podatkovno rudarjenje. Da je dostava vsebin čim bolj uspešna je potrebno obiskovalce segmentirati in prepoznati tiste, ki jih bo določen tip vsebin zanimal. Ena izmed možnosti segmentacije je tudi geografska, kjer posamezna področja (na primer države, regije, občine) razdelimo v take skupine, da se v njih nahajajo čim bolj podobni uporabniki.

[1] `https://en.wikipedia.org/wiki/Ad_serving`

[2] `http://www.zemanta.com/zemanta-programming-challenge-2015/`

To razdelitev lahko nato uporabimo za napovedovanje obnašanja, če poznamo lokacijo uporabnika in imamo pretekle podatke o obiskih in obnašanju.

V diplomskem delu smo iskali skupine, v katere smo združevali poštne številke (območja, ki jih pokrivajo) v ZDA. Država je za potrebe dostave pošte razdeljena na več kot 33.000 poštnih številk, o vsakem območju, ki ga poštna številka pokriva pa je na voljo veliko podatkov, s pomočjo katerih je bila narejena razdelitev. Po dobljenih skupinah in gradnji napovednega modela smo ga preizkusili na učnih in testnih podatkih, ki so jih pri podjetju Zemanta dobili iz svoje platforme.

1.1 Cilji

Cilj naloge je poiskati skupine poštnih številk tako, da se v posamezni skupini nahajajo čim bolj podobni prebivalci. Skupine je potrebno tudi smiselno vizualizirati in dokazati, da so res smiselne. Za iskanje skupin bosta preizkušena algoritma DBScan [13] in hierarhično gručenje[3], ter analiza osnovnih komponent za vizualizacijo [16].

Za iskanje skupin smo uporabili podatke ameriškega statističnega urada Census Bureau[4]. Na voljo so natančni demografski in geografski statistični podatki o posamezni poštni številki, pa tudi o podjetjih in ustanovah, ki se nahajajo na določenem območju, ki ga pokriva.

Skupine smo želeli prikazati na zemljevidu, pri čemer naj bi bilo možno prikazati posamezno skupino, ali vse skupine na enkrat, označene z različnimi barvami.

Za napoved klika smo želeli preizkusiti logistično regresijo[5], naključne gozdove [11], ter združitev večih algoritmov z skladanjem [17]. Pri tem najboljše rezultate pričakujemo pri zadnji metodi, ki združuje prednosti obeh prej omenjenih in tipično dosega najboljše rezultate. Na testnih podatkih želimo doseči površino pod krivuljo ROC [5] okoli 0,75.

[3] `https://en.wikipedia.org/wiki/Hierarchical_clustering`
[4] `http://www.census.gov`
[5] `https://en.wikipedia.org/wiki/Logistic_regression`

Poglavje 2

Opis problema

Glavni cilj naloge je napoved klika na oglas, če poznamo lokacijo obiskovalca, oziroma natančneje poštno številko, ki je določena za območje iz katerega prihaja. Zgraditi moramo napovedni model, ki bo iz učnih podatkov, kjer poznamo lokacijo obiskovalca (poštno številko) in klik na oglas (0 ali 1) na testnih podatkih, kjer je klik skrit, podal verjetnost, da se je klik zgodil.

Če bi napovedovali klik samo na podlagi poštnih številk, bi bili rezultati slabi, saj jih je zelo veliko, s tem pa se zmanjša nabor učnih primerov za posamezno številko. Če pa poštne številke smiselno združimo, v denimo 20 skupin, pa število učnih primerov na skupino močno naraste. Smotrno je namreč predvidevati, da se bodo obiskovalci iz sorodnih lokacij na spletu obnašali podobno.

Poštne številke je torej potrebno združiti v skupine tako, da so prebivalci in sama področja čim bolj podobna. Vsako številko oz. območje, ki ga pokriva moramo za potrebe iskanja skupin opremiti z čim več atributi, ki jo opisujejo. Poleg tega, da skupine poiščemo, je izziv zahteval tudi čim boljšo utemeljitev, zakaj določene poštne številke spadajo v to skupino. Odločili smo se za prikaz na zemljevidu, ki geografske podatke najbolj smiselno prikaže. Za vsako skupino smo tudi izračunali povprečje, ter ga primerjali z posameznimi poštnimi številkami, ter vse to interaktivno prikazali.

Za drugi del naloge sta bila na voljo učni in testni nabor podatkov.

Točnost in kvaliteto napovednega modela smo preverjali z metriko površine pod ROC krivuljo (AUC) na testnih podatkih. Za uspešno rešitev je bilo potrebno pravilno indentificirati skupine, ter na podlagi teh skupin na učnih podatkih zgraditi napovedni model. Testni podatki so bili do konca izziva skriti, zato je bilo potrebno testiranje opraviti na delu učnih podatkov. Za to smo uporabili k-kratno prečno preverjanje [10], tehniko, s katero napovedni model učimo in testiramo na istem naboru podatkov, ne da bi se model pretirano prilagodil učnim podatkom.

Poglavje 3

Odkrivanje skupin

Odkrivanje skupin (angl. *clustering*) je ena od osnovnih tehnik podatkovnega rudarjenja [8]. Iščemo take skupine, kjer je primer iz skupine bolj podoben ostalim v njegovi skupini, kot primerom v ostalih skupinah. Za iskanje obstaja več različnih algoritmov, značilno pa je, da ni absolutno najboljšega [1], zato moramo na danih podatkih testirati več njih, in poiskati tistega, ki najde najboljše skupine.

3.1 Pregled področja in tehnik iskanja skupin

Že od vsega začetka se področje podatkovnega rudarjenja osredotoča na iskanje vzorcev v podatkih [2]. Podatkovno rudarjenje se je razvilo iz strojnega učenja in statistke, z razvojem strojne opreme in zmogljivih računalniških sistemov za obdelavo velikih količin podatkov pa so se odprle številne nove možnosti. Poleg aplikacije principov podatkovnega rudarjenja na standardne podatke (npr. poslovne ipd.) se je uporaba razširila tudi na slike, video in multimedijske vsebine.

Običajno iščemo skupine točk v višjem dimenzionalnem prostoru[1]. Podobnost je definirana kot razdalja med točkami, denimo kot evklidska razdalja[2].

[1] `http://web.stanford.edu/class/cs345a/slides/12-clustering.pdf`
[2] `https://en.wikipedia.org/wiki/Euclidean_distance`

Poleg razdalje med samimi točkami, pa je za razumevanje problema pomembna tudi definicija podobnosti skupin. Če smo prej merili razdaljo med samo dvema točkama, moramo zdaj izmeriti razdaljo med dvema skupinama točk[3]. Najpogostejše mere, ki se za to uporabljajo so povprečna razdalja (angl. *average linkage*), razdalja med najbližjima primeroma (angl. *single linkage*), ali razdalja med najbolj oddaljenima primeroma (angl. *complete linkage*).

Razdaljo med posameznimi primeri moramo oceniti. Ena od možnih ocen, ki smo jo uporabili v diplomski nalogi, je evklidska razdalja. Ta je za n-dimenzionalen prostor in primera p in q določena z:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_i - q_i)^2 + \cdots + (p_n - q_n)^2} \tag{3.1}$$

kjer so p_i in q_i koordinate primera v i-ti dimenziji.

3.2 Podatki

V ZDA statistični urad Census Bureau spremlja množico različnih podatkov in statistik, in jih tudi prosto objavlja za uporabo. Na voljo so tudi statistike za poštne številke, kar smo tudi uporabili v diplomskem delu. Te podatke smo uporabili skupaj z osnovnimi podatki. Za te so s strani podjetja Zemanta bili dani podatki o vrsti in številu podjetij, ki se nahajajo v posamezni poštni številki, ter gostoti poselitve in stopnji brezposelnosti. Za večjo natančnost napovedi in smiselnost najdenih skupin smo poiskali in dodali še demografske podatke o starosti, delež posamezne rase, ter razmerje med moškimi in ženskami.

Pred uporabo smo podatke primerno strukturirali in združili v eno csv[4] datoteko. Velikost datoteke z podatki je obsegala kar 4 GB, zato smo pri obdelavi naleteli tudi na nekaj težav z zmogljivostjo in porabo pomnilnika, ter dolge čase izvajanja kode. Končen format datoteke z zbranimi podatki je

[3] `https://en.wikipedia.org/wiki/Cluster_analysis`
[4] `https://en.wikipedia.org/wiki/Comma-separated_values`

bil podan v obliki, kot jo prikazuje tabela 3.1. Taka oblika podatkov omogoča enostavno računanje razdalj med poštnimi številkami.

Tabela 3.1: Oblika podatkov

Poštna številka	Brezposelnost	Povprečna starost	...
10201	5%	40.5	...
10202	7%	35.5	...
...	...	...	...

3.2.1 Združevanje podatkov

Ko smo pridobili vse želene podatke, jih je bilo potrebno urediti v zgoraj omenjeno obliko in odstraniti nepotrebne atribute. Zaradi velikosti, števila ameriških poštnih številk (33.000) in v nekaterih primerih tudi neučinkovite oblike, v kateri so bili zapisani smo se odločili za uporabo knjižnice Pandas[5], ki je dostopna za programski jezik Python. Knjižnica Pandas uvaja strukturo za shranjevanje podatkov dataframe, ki tudi pri večjem obsegu podatkov omogoča hitro in enostavno združevanje po atributih. Tako smo denimo podatke o nezaposlenosti in populaciji združili z sledečo kodo:

```
merge = pd.merge(df_unemployment, df_population,
how='left', left_on='ZIP', right_on='Zip/ZCTA')
```

S tem smo se izognili tudi težavam, ki so se pojavljale zaradi pomanjkljivih podatkov. Za nekatera področja določeni podatki namreč niso bili na voljo, ali pa so nekatere poštne številke manjkale. Z zgornjo kodo smo obdržali samo tiste poštne številke, ki so se nahajale v viru podatkov o nezaposlenosti. Ker so vhodni podatki manjkali večinoma za bolj periferna območja, ki so bila v testnih podatkih slabo zastopana to ni posebej vplivalo na končno točnost našega modela.

[5] http://pandas.pydata.org/

3.2.2 Shranjevanje podatkov

Pridobljeni podatkovni viri so bili v različnih tekstovnih formatih (tsv, csv, xlsx). Za lažjo implementacijo in možnost izvajanja ter sestavljanja končnega nabora podatkov po delih smo uporabljali csv format zapisa. V prid odločitvi je bila tudi dobra podpora csv datotekam v Pythonu, kar je omogočalo enostavno branje in pisanje datotek.

3.3 Uporabljene metode in izvedba

Iskanja skupin smo se lotili s pregledom primernih metod za iskanje. Odločili smo se za metodi DBScan [13] in hierarhično gručenje[6].

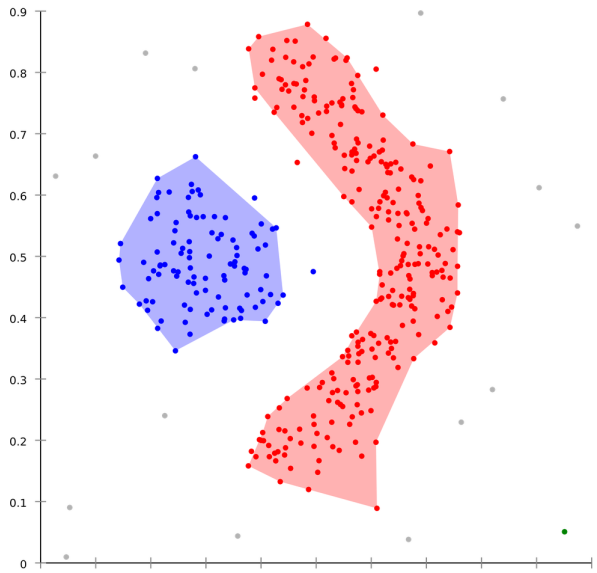
3.3.1 Algoritem DBScan

Algoritem DBScan angl. *density-based spatial clustering of applications with noise* je bil prvič predlagan v članku [13] iz leta 1996. Ideja algoritma je, da okoli izhodiščne točke poišče tiste, ki so v dosegu razdalje ϵ, pri čemer upošteva gostoto točk na območju. Te točke so uvrščene v skupno gručo, ki ji nato pa doda še vse tiste, ki so v dosegu razdalje ϵ na novo dodanih točk.

Algoritem na opisani način upošteva gostoto točk. Kjer je gostota visoka, točke združi v isto skupino, na območjih z nizko gostoto pa točke označi za osamelce. Ali se točka uvrsti v skupino ali pod osamelce uravnavamo z parametrom ϵ. Večja kot je razdalja, manjše je število osamelcev.

Prednost algoritma je, da dobro deluje tudi na podatkih, pri katerih ostali algoritmi, npr. hierarhično gručenje ali metoda voditeljev ne najdejo pravih gruč. Primer takih podatkov, kjer DBScan pravilno najde gruči, metoda voditeljev pa ne je prikazan na sliki 3.1. Drugi prednosti sta še, da ni potrebno vnaprej določiti števila gruč, ter da zna pravilno določiti osamelce. Osamelci so tisti primeri, ki so preveč različni od ostalih gruč, da bi bili uvrščeni v katero izmed njih.

[6] `https://en.wikipedia.org/wiki/Hierarchical_clustering`



Slika 3.1: Primer pravilno določenih gruč z DBScan algoritmom

Slabosti algoritma DBScan so nevarnost uvrstitve vseh primerov v eno gručo, ali pa vseh primerov med osamelce. Prav s to pomanjkljivostjo algoritma smo se srečali pri analizi naših podatkov.

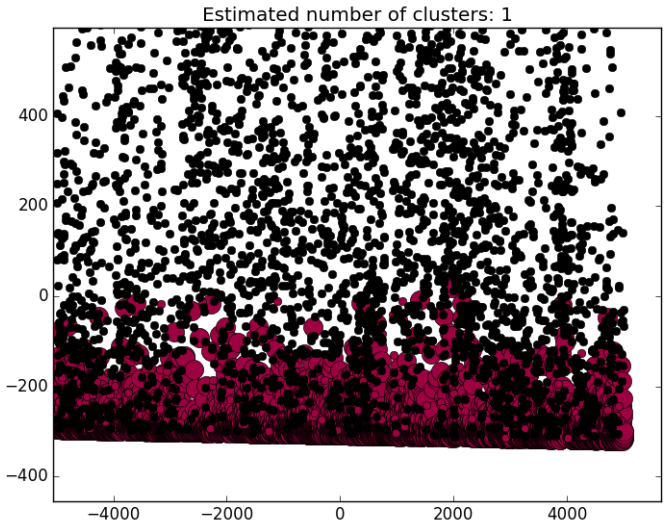
Izvedba

Uporabili smo implementacijo algoritma DBScan, ki je dostopna v knjižnici Scikit learn[7]. Uporaba knjižnic nam poenostavi kodo, hkrati pa tudi izboljša hitrost programa, saj so algoritmi dobro optimizirani. Primer klica algoritma nad podatki v spremenljivki data podaja spodnja koda, ki najdene gruče shrani v spremenljivko labels:

```
db = DBSCAN(eps=10, min_samples=10).fit(data)
labels = db.labels_
```

V namene vrednotenja najdenih skupin smo dobljene gruče grafično prikazali, pri čemer pa smo morali najprej več dimenzionalni prostor preslikati v dvodimenzionalnega. Uporabili smo analizo osnovnih komponent, o kateri več v nadaljevanju. Grafični prikaz rezultatov gručenja je pokazal prve težave

[7] http://scikit-learn.org/



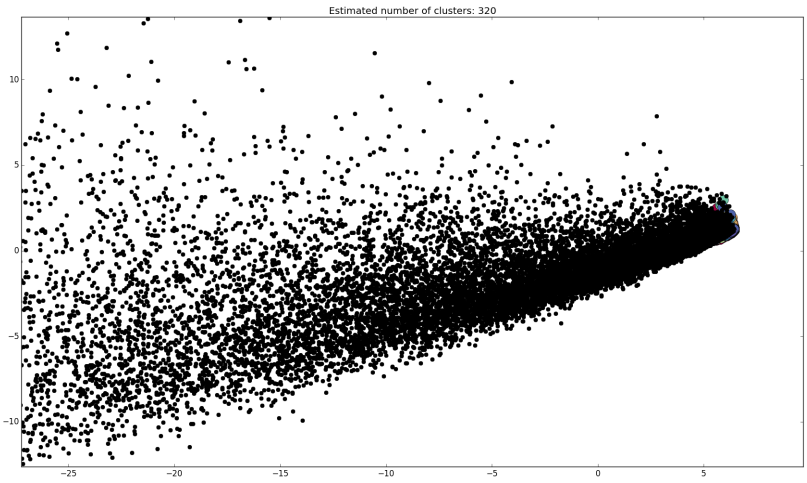
Slika 3.2: Primer neustreznega gručenja, kjer je najdena le ena gruča (vijolično), ostali primeri pa so uvrščeni kot osamelci

algoritma DBScan. Glede na razdaljo, ki smo jo določili (parameter *eps=10* v zgornji kodi), sta se dogajala dva scenarija.

V prvem (slika 3.2), kjer je bila razdalja, pri kateri je primer še spadal v isto gručo večja, so bili vsi primeri uvrščeni v eno ali dve gruči, ostali primeri pa so bili uvrščeni kot osamelci. Tak rezultat je bil že intuitivno neustrezen, saj lahko brez zadržkov trdimo, da v ZDA prebivalci živijo v več kot enem ali dveh različnih okoljih. Težava je torej bila, da je algoritem zaradi prevelike razdalje uvrščal v isto gručo, zato smo se odločili razdaljo zmanjšati.

Zmanjšanje razdalje je prineslo novo težavo. Število gruč je nenadoma preseglo smiselne meje, saj je bilo najdenih več kot 300, pri čemer se je ne glede na večjo eps razdaljo močno povečalo število osamelcev, kar je razvidno iz slike 3.3.

Izkazalo se je, da glede na vhodne podatke, ki smo jih imeli na voljo DBScan ni primerna metoda za iskanje gruč.



Slika 3.3: Primer izvedbe DBScan z manjšo eps razdaljo, pri čemer je bilo najdenih 320 gruč, osamelci pa močno prevladujejo. Gruče se nahajajo samo na skrajni desni.

3.3.2 Hierarhično gručenje

Hierarhično gručenje je pogosto uporabljan za iskanje gruč. Gradi hierarhijo med posameznimi gručami. Ločimo dva pristopa. Pri prvem algoritem gradi gruče od spodaj navzgor in jih združuje v končno gručo, medtem ko pri drugem pristopu iz začetne, ki zajema vse primere rekurzivno deli v manjše gruče. Nastalo hierarhijo lahko prikažemo z dendrogramom[8].

Kot že omenjeno v poglavju Pregled področja in tehnik iskanja skupin, je pri hierarhičnem gručenju pomembna izbira metoda za izračun razdalje med skupinami. Preizkusili smo povprečno[9] (angl. *average linkage*) in wardowo razdaljo[10] (angl. *ward linkage*).

- Povprečna razdalja med dvema skupinama A in B je izračunana tako, da vzamemo povprečje vse razdalj med pari točk (a in b) iz teh dveh

[8] `https://en.wikipedia.org/wiki/Dendrogram`

[9] `https://en.wikipedia.org/wiki/Hierarchical_clustering`

[10] `http://sites.stat.psu.edu/~ajw13/stat505/fa06/19_cluster/09_cluster_wards.html`

skupin:

$$d(A,B) = \frac{1}{|A||B|} \sum_{a \in A} \sum_{b \in B} d(a,b) \tag{3.2}$$

- Wardova razdalja združi gruče združuje tako, da minimizira vsoto kvadratov razlik med njimi. Hierarhično gručenje ima težnjo, da večje gruče postajajo še večje, kar lahko pripelje do rezultatov, kot smo jih dobili pri DBScan. To težavo delno omili izbira Wardove razdalje, kar se je izkazalo tudi na naših podatkih.

$$d_{ij} = d(\{X_i\}, \{X_j\}) = \|X_i - X_j\|^2 \tag{3.3}$$

Pri izvedbi hierarhičnega gručenja z uporabo povprečne razdalje smo naleteli na podobne težave kot pri algoritmu DBScan, kar je razvidno iz slike 3.4. Namesto osamelcev smo dobili prevladujočo gručo, ostale pa so bile zastopane minimalno. Pri hierarhičnem gručenju je potrebno število skupin, ki naj jih algoritem poišče omejiti, zato smo se odločili za 20 skupin oz. gruč.

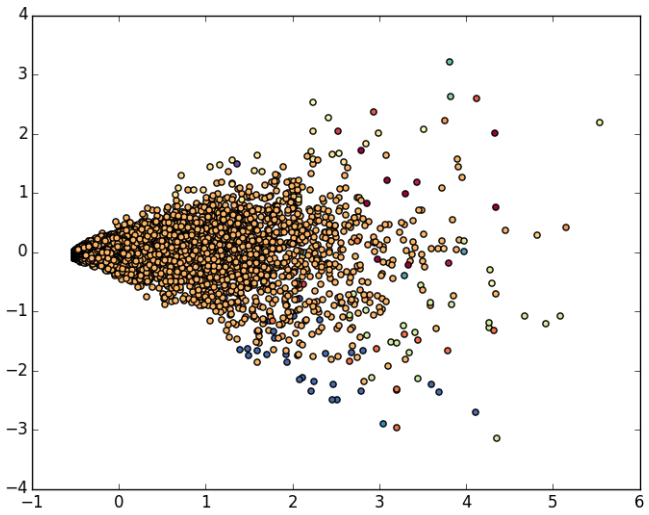
V naslednjem poskusu smo uporabili Wardowo razdaljo, s katero smo pridobili najbolj smiselne rezultate, ki so bili tudi uporabljeni za napovedovanje. Gruče so bile bolj enakomerno zastopane in so se najbolj približale rezultatu, ki smo ga intuitivno pričakovali.

Ponovno smo uporabili Python knjižnico Scikit-learn, s pomočjo katere je implementacija gručenja enostavna:

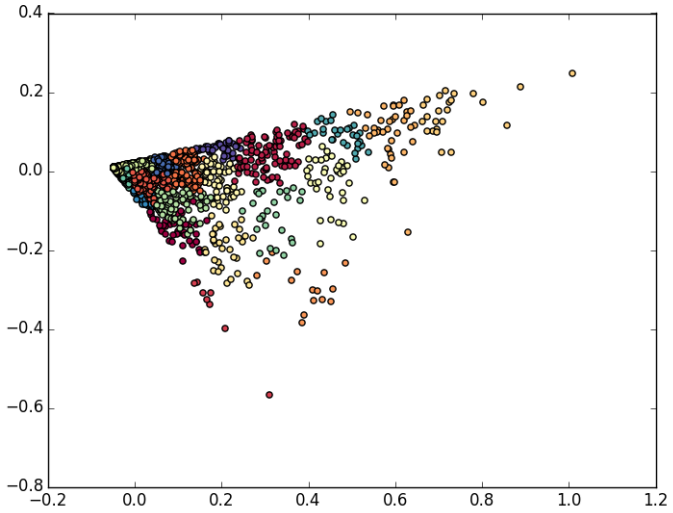
```
hc = AgglomerativeClustering(n_clusters=40,
linkage='ward').fit(data)
```

V spremenljivko *hc* se shrani oznaka skupine kateri pripada za vsak vhodni primer. Tako je bila vsaka izmed 33,000 poštnih številk z pripadajočimi atributi uvrščena v eno izmed 20 skupin.

Sama izvajanje kode je bilo dolgotrajno (več ur), saj smo imeli opravka z velikim številom primerov (vse poštne številke), pa tudi z velikim številom atributov za vsak primer. To je tudi nekoliko otežilo gručenje z različno



Slika 3.4: Hierarhično gručenje z povprečno razdaljo.



Slika 3.5: Hierarhično gručenje z Wardowo razdaljo.

omejitvijo števila gruč, ki naj jih algoritem najde, saj sta večkratno izvajanje in primerjava rezultatov vzela veliko časa.

3.4 Prikaz in interpretacija skupin

Prikaz rezultatov v podatkovnem rudarjenju predstavlja svojevrsten izziv, saj se pogosto srečujemo z več dimenzionalnim prostorom, ki ga je potrebno smiselno prikazati. Posledično so se razvile številne tehnike, ki omogočajo preslikavo iz več dimenzionalnega prostora v 2 ali 3 dimenzije, ki jih nato lahko prikažemo. Ena izmed takih tehnik je analiza osnovnih komponent [16], ki smo jo uporabili za vizualizacijo rezultatov hierarhičnega gručenja na slikah 3.2, 3.3, 3.4 in 3.5.

Ker pa vse skozi govorimo o geografski segmentaciji, poštne številke pa imajo točno določeno lokacijo smo podatke prikazali tudi na zemljevidu s pomočjo zemljevidov Google Earth[11].

3.4.1 Vizualizacija z analizo osnovnih komponent

Analiza osnovnih komponent [16] (angl. *principal component analisys*, PCA) je metoda, s katero poiščemo linearno preslikavo iz večdimenzionalnega prostora v nekaj dimenzionalni prostor. V našem primeru je cilj dvodimenzionalna slika, ki prikazuje najdene gruče.

Analiza osnovnih komponent išče vektorje oz. osnovne komponente, ki pojasnijo čim več variance v podatkih. Če imamo 100 dimenzionalni prostor, bi 100 komponent popolnoma pojasnilo varianco podatkov. Ker pa nas zanima približna preslikava v nižje dimenzije pa je dovolj, da vzamemo samo prvi dve komponenti, ki pojasnita največ variance:

```
colors = plt.cm.Spectral(np.linspace(0, 1,
len(unique_labels)))
i_pca = IncrementalPCA(n_components=2,
batch_size=10000)
X = i_pca.fit(data).transform(data)
```

[11] https://www.google.com/earth/

```
print "done---"
for i in range(len(X)):
    plt.scatter(X[i, 0], X[i, 1], c=colors[labels[i]])
plt.show()
```

Zaradi velikega števila poštnih številk, nad katerimi smo izvajali hierarhično gručenje, smo naleteli na težave z porabo pomnilnika pri izvedbi analize osnovnih komponent. Kot je razvidno iz zgornje kode smo zato uporabili inkrementalno izvedbo PCA, ki vhodne podatke obdeluje po delih, (batch size), v našem primeru 10.000 hkrati.

V spremenljivko *colors* smo najprej shranili toliko različnih barv, kolikor je bilo najdenih gruč. Nato smo izvedli PCA in nazadnje v zanki še izrisali vsak primer v ustrezni barvi.

3.4.2 Prikaz rezultatov na zemljevidu

Grafični prikaz točk z analizo osnovnih komponent je služil predvsem kot pomoč za lažje razumevanje rezultatov gručenja. Same najdene skupine pa smo grafično utemeljili z prikazom na zemljevidu, ki se je izkazal za zelo informativnega in smiselnega.

Ker je bilo potrebno prikazati vse poštne številke smo pri večini spletnih servisov (Google Maps, Bing Map) naleteli na omejitve, ki veljajo za prikaz večjega števila lokacij na zemljevidu. Omejitve veljajo tudi za kodiranje naslovov v koordinate. Preden lahko posamezen naslov (ali poštno številko) prikažemo, moramo najprej pridobiti njene koordinate. To omogočajo omenjeni ponudniki, a v precej manjšem dnevnem obsegu kot za naše potrebe.

Omejitve smo zaobšli z uporabo Google Earth z KML datotekami, pri čemer smo na spletu pridobili bazo ameriških poštnih številk z že podanimi koordinatami. S tem smo zaobšli vse omejitve, ki so se pojavile zaradi velike količine lokacij.

KML zapis podatkov

datoteka KML[12] opisuje geografske podatke, ki jih želimo prikazati v za to namenjenih aplikacijah, kot je denimo Google Earth[13]. Gre za prilagojeno obliko bolj znanega XML zapisa. Definirane so značke, s katerimi določimo izgled in lokacijo na zemljevidu. Odločili smo se za prikaz posamezne poštne številke z t. i. žebljičkom, kjer je pripadnost posamezni skupini določena z barvo.

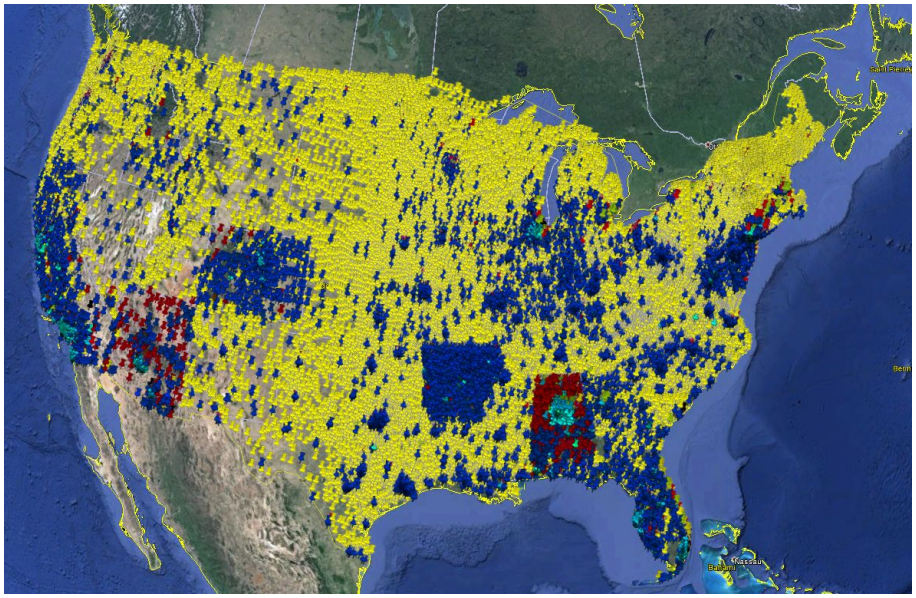
```
      <Placemark>
    <ExtendedData>
      <Data name="ZIP">
        <value>35004</value>
      </Data>
      <Data name="CLUSTER␣STATS">
        <value>/</value>
      </Data>
      ...
    <Point>
      <coordinates>-86.50249,33.606379</coordinates>
      <color>#CC9900</color>
    </Point>
    <styleUrl>#4</styleUrl>
  </Placemark>
```

Zgornji izsek iz datoteke KML definira lokacijo s koordinatami, ki smo jih pridobili na spletu. Z `<styleURL>` značko je določena barva in oblika žebljička, `<ExtendedData>` pa opisuje vrednosti, ki so prikazane na oknu, ki se prikaže ob kliku na posamezen žebljiček.

Gradnja datoteke KML

Tudi za gradnjo datoteke KML smo uporabili Python. Za osnovo nam je služila koda podana iz strani Googla[14], ki smo jo prilagodili za naše potrebe. V prvem koraku smo izračunali povprečja za vsako najdeno skupino. Nato smo iz datoteke z koordinatami vsaki poštni številki dodelili zemljepisno širino in dolžino. Glede na oznako skupine, kateri je posamezna poštna

[12] `https://developers.google.com/kml/`
[13] https://www.google.com/earth/
[14] https://developers.google.com/kml/articles/csvtokml?csw=1



Slika 3.6: Rezultat gručenja, prikazan na zemljevidu. Pripadnost skupini je označena z barvo.

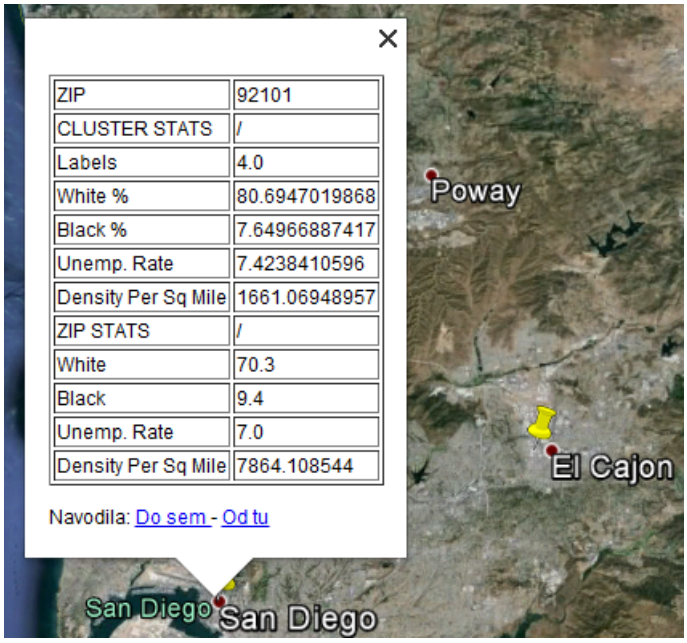
številka oz. točka na zemljevidu pripadala je bila določena barva, na okno, ki se prikaže ob kliku pa smo dodali izračunana povprečja in dejanske podatke za poštno številko, kar je olajšalo evaluacijo rezultatov.

Končen rezultat je datoteka KML, katero lahko odpremo v programu kot je Google Earth, ki poskrbi, da se poštne številke prikažejo na zemljevidu.

Google Earth

Uvoz datoteke v program Google Earth da sledeč rezultat 3.6. Hitro lahko prepoznamo najbolj pogosto skupino, ter ocenimo smiselnost gručenja. Pripravili smo tudi KML datoteke za posamezno skupino, da je bila analiza skupin lažja.

Ob izboru posameznega žebljička se prikažejo vsi podatki za poštno številko, kot prikazuje slika 3.7.



Slika 3.7: Informacije o posamezni poštni številki in povprečje skupine, kateri pripada.

3.5 Rezultati iskanja skupin

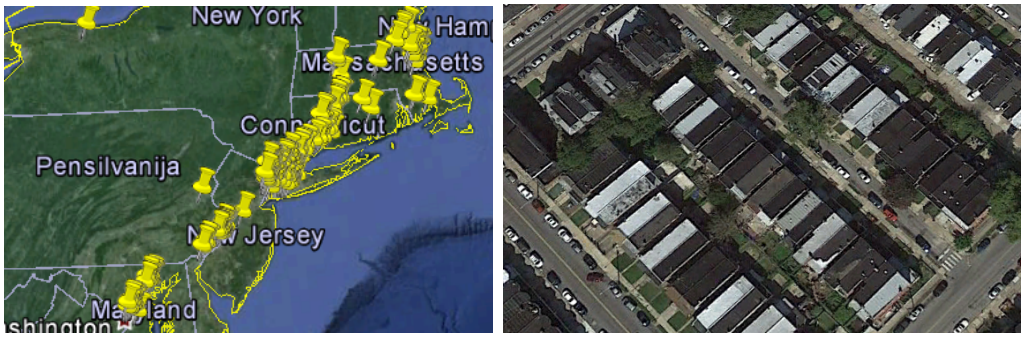
Ko smo bili z rezultati gručenja in vizualizacijo zadovoljni, smo dobljene skupine natančneje analizirali. Izbrali smo pet najbolj izstopajočih skupin in jih natančneje opisali, ostale skupine pa smo samo poimenovali, glede na najbolj izstopajočo lastnost. Opisi skupin so podkrepljeni z izseki iz grafičnega prikaza na zemljevidu.

3.5.1 Najznačilnejše skupine

Spodaj naštejemo in opišemo najznačilnejše skupine, ki smo jih identificirali iz podatkov.

Skupina 1, okolica centrov velikih mest

Za skupino je značilna nizka brezposelnost, visok odstotek temnopoltih prebivalcev, nizka povprečna starost in povprečni prihodki. Iz grafičnega prikaza na zemljevidu je razvidno, da je center mesta vedno v bližini, prebivalci pa živijo v hišah ali v večstanovanjskih zgradbah. Skupina je dobro zastopana in zelo homogena.



Slika 3.8: Naselja hiš, značilnih za skupino 1.

Skupina 3, velemesta, srednji razred

V tej skupini najdemo ameriška velemesta (New York, Miami, Chicago, Phoenix ipd.), značilna je visoka gostota prebivalstva, ki večinoma živi v stolpnicah. Povprečna gostota poselitve je 57 000 prebivalcev na kvadratno miljo. Prihodki so nizki, zelo visok je odstotek temnopoltega prebivalstva, saj na področjih, ki jih pokrivajo poštne številke pripadajoče tej skupini presegajo 50 odstotkov.

Skupina 10, prostrana ruralna območja na jugu ZDA

Večino poštnih številk v tej skupini najdemo na jugu ZDA v bližini mehiške meje. Povprečno število prebivalcev je nizko, gostota poselitve je manjša od 100 prebivalcev na kvadratno miljo. Značilna je še visoka brezposelnost, ki se giblje okoli 10 odstotkov in velike površine, ki jih posamezne poštne številke pokrivajo - preko 500 kvadratnih milj.

Slika 3.9: Chicago in New York - vsi primeri se nahajajo v centru mest.

Slika 3.10: Prostrana območja v bližini mehiške meje.

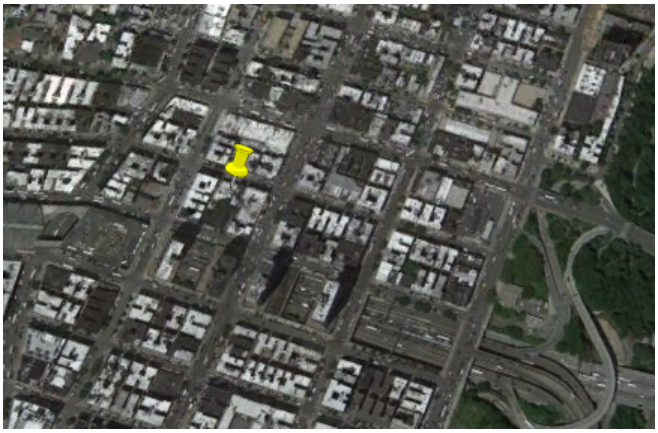
Skupina 7, gosto poseljena, rasno mešana območja.

Visoka gostota poselitve, in nizek odstotek belopoltih prebivalcev sta značilnost skupine. Območja so večinoma locirana v velikih mestih. Povprečna brezposelnost za skupino je 9 odstotna, kar je nad ameriškim povprečjem[15] (5.6 odstotka).

Skupina 8, ameriški višji razred, urbane soseske

Vile z bazeni so tipični pogled, ki ga dobimo ob približanju posameznega področja na zemljevidu, ki pripada tej skupini. Urban življenski slog v pred-

[15]http://www.bls.gov/news.release/empsit.nr0.htm



Slika 3.11: Na področju, ki ga pokriva ta poštna številka živi več kot 100.000 ljudi.

mestjih, z visokimi prihodki in nekoliko starejšim prebivalstvom.

Slika 3.12: Tipična vila z bazenom, v skupini 8.

3.5.2 Ostale skupine

Preostale skupine, ki niso med zgoraj naštetimi, so:

- Skupina 0, ruralna območja z visoko brezposelnostjo

- Skupina 2, prostrana območja Aljaske
- Skupina 4, predmestna naselja
- Skupina 5, kmetije in kmetijske površine
- Skupina 6, mešano območje, prevladuje podeželje
- Skupina 9, manjše vasi na podeželju
- Skupina 10, nizko poseljena prostrana območja
- Skupina 11, predmestja
- Skupina 12, mesta na zahodni obali
- Skupina 13, podeželje, 90 odstotni delež belopoltih
- Skupina 14, rasno mešana urbana območja z prevladujočim temnopoltim in latino prebivalstvom
- Skupina 15, predmestja
- Skupina 16, industrijska območja
- Skupina 17, mesta Alabame
- Skupina 18, trgovski centri
- Skupina 19, bogate soseske z nizko brezposelnostjo

Poglavje 4

Napovedovanje

Napovedovanje[1] (angl. *Predictive modelling*) je področje podatkovnega rudarjenja, ki se ukvarja z napovedovanjem bodočih dogodkov na podlagi podatkov o dogodkih, ki so se že zgodili. Tako je možno napovedati, kolikšna je verjetnost za ponovno pojavitev zdravstvenih težav, s čimer so v bolnišnici Parkland Health and Hospital System[2] v Dallasu zmanjšali število bolnikov, ki jih je potrebno ponovno sprejeti v bolnišnico za 33 odstotkov.

Napovedovanje je poleg zdravstva množično uporabljeno tudi v prodajnem sektorju, spletnih aplikacijah, biologiji, financah, zavarovalništvu, ter tudi v oglaševanju, s čimer se ukvarjamo v diplomski nalogi, kjer je cilj čim bolj natančno napovedati verjetnost klika na oglas. V praksi nam to daje možnost ciljnega oglaševanja, kjer oglas prikažemo tistim obiskovalcem, za katere je napovedana najvišja verjetnost klika.

[1] `http://www.gartner.com/it-glossary/predictive-modeling`

[2] `http://healthleadersmedia.com/page-1/FIN-279439/How-Predictive-Modeling-Cuts-Hospital-Readmissions`

4.1 Pregled področja in metod napovedovanja

Pregled področja in metod napovedovanja povzemamo po Jiawei in Kamber [4]. Pri napovedovanju ločimo dva osnovna principa - uvrščanje oz. klasifikacijo in napovedovanje zveznih vrednosti.

4.1.1 Uvrščanje

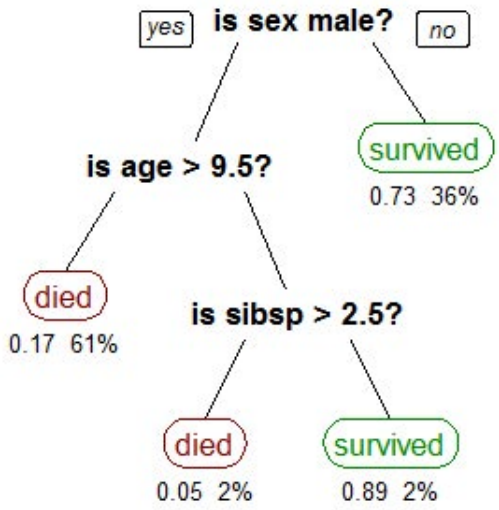
Pri uvrščanju so razredi, v katere lahko primeri spadajo, že določeni. Če glede na simptome napovedujemo bolezen, gre za uvrščanje, saj imamo končno množico možnih bolezni, v katere posamezen primer lahko uvrstimo. Na podlagi atributov, ki posamezen primer opisujejo torej določamo razred. Učni podatki pri uvrščanju vsebujejo primere, opisane z atributi, ter razred, v katerega spadajo. Na podlagi učnih podatkov zgradimo model, ki zna v enega izmed razredov bolj ali manj natančno uvrstiti tudi nove primere (testni podatki), pri katerih razred ni podan, oz. je skrit.

Pri uvrščanju je zelo pomembno kateri atribut je najbolj informativen, kar pomeni, da je njegov prispevek najvišji za uvrstitev v določen razred. Za določanje informativnosti atributa se uporablja več mer, denimo informacijski prispevek, relativni informacijski prispevek ali ginijev indeks.

Preprosto orodje s katerim lahko gradimo napovedne modele za uvrščanje v razrede so klasifikacijska drevesa[3]. Na vsaki vejitvi drevesa začetno množico podatkov razbijemo na dve novi podmnožici, kar nas pripelje do čistih podmnožic. Večji informacijski prispevek ima atribut, višje v drevesu ga uporabimo za odločanje, kako bomo razdelili primere na podmnožice.

Pogosto uporabljane metode za uvrščanje so še naključni gozdovi [11], kjer združimo več dreves, ki glasujejo glede uvrstitve primera v razred, metoda podpornih vektorjev [9] in metoda k najbližjih sosedov [14].

[3] `https://en.wikipedia.org/wiki/Decision_tree_learning`



Slika 4.1: Primer klasifikacijskega drevesa, ki uvršča v dva razreda.

4.1.2 Napovedovanje zveznih vrednosti

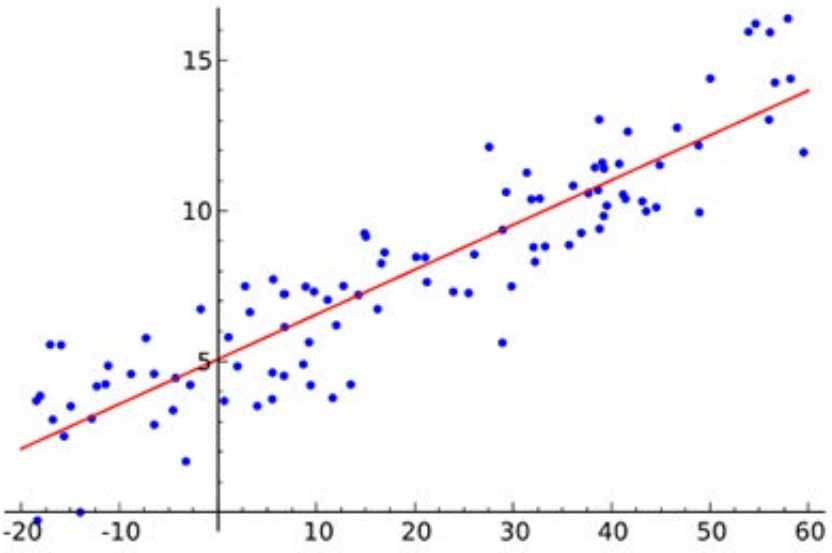
Pri napovedovanju zveznih vrednosti razred pri učnih razredih ni podan, pač pa je podana zvezna vrednost, ki jo posamezen primer zavzema. Tudi pri napovedovanju torej ne uvrščamo v enega izmed določenih razredov, ampak glede na atribute novega primera napovemo zvezno vrednost. Uporabljamo tehnike regresijske analize, kot sta linearna in logistična regresija.

Pri linearni regresiji[4] tako zgradimo model, ki ga lahko predstavimo s premico na sliki 4.2. Nove vrednosti so preslikane v skladu z funkcijo, ki določa premico.

4.1.3 Pretirano prilagajanje učnim podatkom

Pretirano prilagajanje učnim podatkom [7] (angl. *overfitting*) je težava, ki se pojavi, ko se model pretirano prilagodi učnim podatkom, kar posledično

[4] `https://en.wikipedia.org/wiki/Linear_regression`



Slika 4.2: Linearna regresija.

pomeni nizke napovedne točnosti na testnih podatkih. Brez težav namreč zgradimo model, ki se bo popolnoma prilagodil učnim podatkom, seveda pa bi takšen model na testnih podatkih praviloma dosegal zelo slabe rezultate.

Želimo si torej modela, ki se iz podatkov uči, ne pa da si jih skuša zapomniti[5]. Informacije, ki jih algoritem zna pridobiti iz učnih podatkov, lahko namreč razdelimo na dva tipa. Tiste, ki bodo imele vpliv na prihodnost in tiste informacije, ki so specifične in pri napovedovanju prihodnosti pomenijo šum.

Pomembno vlogo pri pretiranem prilagajanju igra kompleksnost modela. Bolj ko je model kompleksen, na manj primerih se ta nauči vrednost svojih parametrov. Želimo si torej čim bolj preprostih modelov, z majhnim številom parametrov, saj s tem zmanjšamo nevarnost pretiranega prilagajanja. Če se model uspešno izogne tem težavam, pravimo da je robusten.

V namene detekcije prevelikega prileganja uporabljamo testiranje točnosti modelov s prečnim preverjanjem [10]. Tehnike, ki nam pri gradnji modelov

[5] `https://en.wikipedia.org/wiki/Overfitting`

omogočajo gradnjo preprostejših modelov, so regularizacija[6] in rezanje dreves[7].

4.2 Podatki

Učni podatki so bili podani iz strani podjetja Zemante. Pridobili so jih iz njihove oglaševalske platforme. Gre torej za realne podatke o obiskovalcih spletnih strani njihovih strank. Testni podatki so bili do konca izziva skriti, ter so bili uporabljeni za evaluacijo naših rešitev. Za interno preverjanje točnosti naših modelov smo, kot je pri takšnih tekmovanjih praksa, morali uporabiti učne podatke, o čemer več v poglavju 4.4.

Oblika podatkov je podana na primeru v tabeli 4.1. V stolpcu *click* je zabeleženo, ali je obiskovalec, ki mu je bil oglas prikazan kliknil nanj. Creative id je številka, dodeljena posameznemu oglaševalcu iz strani Zemantinega sistema in za naše potrebe nima večje vloge. V naslednjem stolpcu se nahaja poštna številka, iz katere je bil obiskovalec, pri čemer je potrebno dodati, da je ta številka pogosto manjkala, zato smo manjkajoče vrednosti zamenjali z ničlo. Podana sta bila še stolpca domena in stran, ki sta povedala, na kateri domeni in strani je bil oglas prikazan.

Učni podatki so obsegali približno 2,5 milijona vrstic, zate se pomanjkanja primerov ni bilo bati. So se pa posledično pojavile nekatere performančne težave, predvsem dolgi časi izvajanja bolj kompleksnih modelov. Za večjo natančnost napovedi smo obstoječim podatkom dodali stolpec skupina, v kateri smo vsaki poštni številki dodali informacijo o skupini, kateri je glede na rezultate prvega dela naloge spadala. S tem se je vrednost oz. informativnost lokacije povečala, saj smo vsako izmed 33.000 poštnih številk opisali z eno od 20 najdenih skupin.

Za pravilno pripenjanje podatka o pripadnosti skupini za poštne številke je skrbela sledeča koda:

```
    with open(labels_file, mode='r') as file_in:
```

[6] https://en.wikipedia.org/wiki/Regularization_(mathematics)
[7] https://en.wikipedia.org/wiki/Pruning_(decision_trees)

```
        reader = csv.reader(file_in)
        c_labels = {float(rows[0]): rows[1] for
        rows in reader}
    #change ZIP with label
    l_set['zip'] = l_set['zip'].convert_objects
    (convert_numeric=True).dropna()
    l_set['zip'] = l_set['zip'].map(c_labels.get)
```

Podatki o pripadnosti skupini so bili zapisani v datoteki *labels_file*, stolpec ZIP, ki je vseboval poštne številke pa smo nato z funkcijo *map*[8] zamenjali poštno številko z skupino.

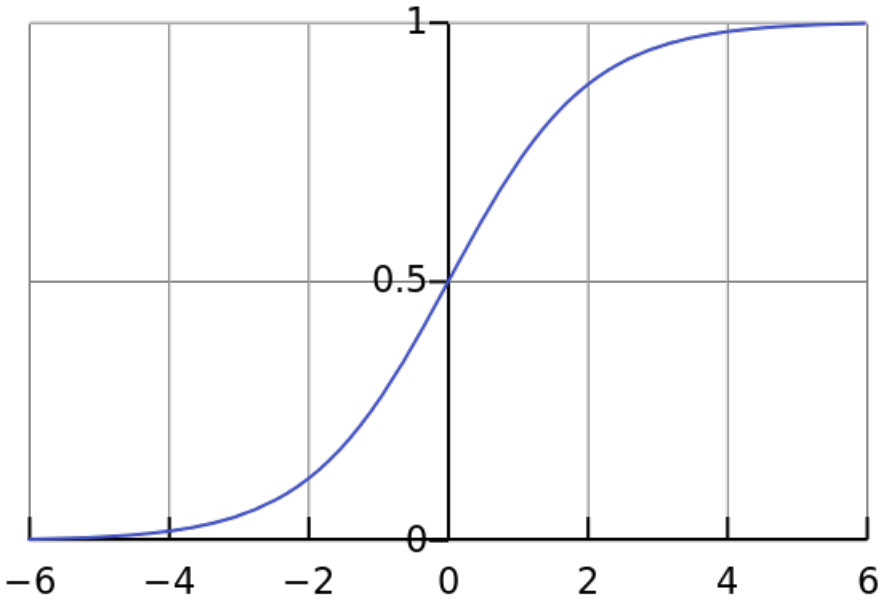
Tabela 4.1: Oblika učnih podatkov

click	creative id	ZIP	domain	page
1	2522	70611	townhall.com	http://townhall.com/
0	64	98022	twitchy.com	http://twitchy.com/
0	2522	44646	townhall.com	http://townhall.com/
1	2380	43230	allday.com	http://allday.com/post/2533..
...	...	...	...	...

4.3 Uporabljene metode in praktična izvedba

Pri napovedovanju vrednosti smo morali na podlagi učnih primerov napovedati verjetnost klika, torej uvrstitve v razred 1. Gre torej za klasifikacijski problem, kjer pa nismo napovedovali razreda, oz. ali se bo klik zgodil ali ne, temveč verjetnost tega dogodka. Odločili smo se za preizkus algoritmov logistične regresije, naključnih gozdov in tehnike skladanja (angl. stacking), kjer združimo več napovednih modelov. Vsi algoritmi so bili implementirani z knjižnico Scikit Learn.

[8] `https://docs.python.org/2/library/functions.html`



Slika 4.3: Graf logistične funkcije.

4.3.1 Logistična regresija

Logistična regresija[9] je regresijski model, s katerim običajno napovedujemo binarno odvisno spremenljivko. To pomeni, da napovedujemo le dva razreda, tako kot v našem primeru. Obstaja tudi možnost napovedi več razredov, ki pa za naš primer ni aktualna.

Osnova za logistično regresijo je logistična funkcija (4.1), s katero ocenjujemo verjetnost kategorične spremenljivke. Logistična funkcija ne glede na vhod vedno vrne rezultat med 0 in 1.

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}} \tag{4.1}$$

V našem primeru je logistična regresija služila za združevanje napovedi ostalih modelov pri skladanju.

[9] https://en.wikipedia.org/wiki/Logistic_regression

4.3.2 Naključni gozdovi

Metoda naključnih gozdov [11] (angl. *Random forest*) spada med sestavljene metode (angl. *ensemble methods*), ki združujejo rezultate več različnih klasifikatorjev. V primeru naključnih gozdov združujemo napovedi posameznih dreves, lahko rečemo, da skupina dreves, ki sestavlja gozd o vsakem primeru glasuje. Naključni gozd primer uvrsti v razred, ki ga napove večina dreves. Kako izgleda posamezno drevo lahko vidimo na sliki 4.1. Uporablja se lahko tako za regresijo kot za klasifikacijo. Naključni gozdovi odpravljajo največjo težavo odločitvenih dreves, pri katerih hitro pride do pretiranega prilagajanja učnim podatkom, pojem ki smo ga obravnavali v podpoglavju 4.1.3.

Želimo si čim bolj različnih dreves, s katerimi bomo našli tudi kakšno specifiko podatkov, ki jih posamezno drevo ne zazna. Če bi vsako drevo zgradili iz vseh dostopnih podatkov, bi dobili enaka drevesa, kar ne bi imelo smisla. Zato drevesa gradimo z metodo stremena (angl. *bootstrap*), kjer za vsako drevo naključno z vračanjem iz učne množice izberemo podmnožico primerov. Na tej podmnožici v naslednjem koraku zgradimo odločitveno drevo, pri čemer v vozliščih drevesa ne upoštevamo vseh atributov, ampak naključno izberemo manjše število le-teh.

Z omenjenim postopkom pridobimo različna drevesa, katerih napovedi združimo. Pri klasifikaciji upoštevamo večinsko napoved, pri regresiji pa vzamemo povprečje napovedi.

Naključni gozdovi so ena izmed najbolj točnih metod uvrščanja. Slabosti metode sta, da v primerjavi s posameznimi drevesi ne moremo interpretirati modela, poveča pa se tudi zahtevnost izvedbe.

Izvedba

Knjižnica Scikit Learn vključuje tudi implementacijo naključnih gozdov[10]. Vnaprej se je potrebno odločiti za število dreves, ki bodo glasovala o uvrstitvi v razred. Odločili smo se za 100 dreves, saj je manjše število dreves vodilo

[10] `http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

v padanje točnosti, povečevanje pa v težave z zmogljivostjo in dolge čase izvedbe algoritma.

```
X_train, X_test, y_train, y_test =
cross_validation.train_test_split(X, y, test_size=0.4)
forest = RandomForestClassifier(n_estimators=n_estimators)
#fit training data
prob = forest.fit(X_train, y_train,).predict_proba(X_test)
#compute ROC
fpr, tpr, thresholds = roc_curve(y_test, prob[:, 1])
roc_auc = auc(fpr, tpr)
print roc_auc
```

Z zgornjo kodo smo implementirali napoved verjetnosti razreda. Vhodne podatke smo razdelili na učni in testni set, da smo lahko takoj vrednotili točnost naše implementacije. Spremenljivka *forest* hrani objekt iz knjižnice. V naslednji vrstici pridobimo verjetnosti razreda 1 za testne primere, saj z metodo *fit()* model naučimo na učnih podatkih, z *predict_proba(X_test)* pa že napovemo verjetnosti razredov za testne podatke.

Z metodo *roc_curve()* iz knjižnice *scikit.metrics* izračunamo delež napačno pozitivnih in zadetih primerov, kar je podlaga za izračun AUC v naslednji vrstici, o čemer več v poglavju 4.4.

4.3.3 Skladanje

Skladanje [15], [17] (angl. *stacking oz. stacked generalization*) je metoda, pri kateri združujemo več različnih napovednih modelov. V prvem koraku zberemo napovedi vseh modelov na nivoju 0 v nov nabor podatkov, ki jih skupaj z napovedjo razreda in dejanskim pravilnim razredom obravnavamo kot nov odločitveni problem, za rešitev katerega uporabimo model na nivoju 1, navadno logistično regresijo. Gre za metodo, ki običajno daje najboljše rezultate, z njeno izpeljanko pa je bila dosežena tudi zmaga na prestižnem Netflixovem tekmovanju[11].

V našem primeru smo na nivoju 0 uporabili sledeče napovedne modele: Random Forest (Naključni gozdovi), Extra Trees[3] (Ekstremno naključna

[11] http://bits.blogs.nytimes.com/2009/09/21/netflix-awards-1-million-prize-and-starts-a-ne
?ref=technology&_r=1

drevesa) in Gradient Boosting. Random Forest oz. naključne gozdove smo že predstavili, Extra trees pa mu je zelo podoben. Razlikujeta se pri gradnji posameznih dreves, ki so v primeru Extra trees bolj naključna zaradi načina delitve v listih. S tem dobimo nov pogled na podatke, je pa metoda zaradi tako velike naključnosti primerna samo za uporabo pri skladanju, ne pa kot samostojna metoda. Tudi Gradient Boosting[12] sestavlja množica dreves, ki jih algoritem združi z optimiziranjem cenilne funkcije.

Napovedi omenjenih treh modelov smo združili z logistično regresijo, ki smo jo natančneje opisali v poglavju 4.3.1.

Izvedba

Implementacija skladanja temelji na primeru[13] in je bila ustrezno prilagojena za napovedovanje verjetnosti razreda. Podatke smo najprej razdelili na učni in testni nabor, pri čemer smo za testiranje namenili 20 odstotkov podatkov.

```
    X_train, X_test, y_train, y_test =
    cross_validation.train_test_split(X, y, test_size=0.2)
```

Učne primere smo razdelili na pet delov. Za vsak klasifikator in za vseh 5 delov na katerega so bili razdeljeni podatki (5-kratno prečno preverjanje) smo vsakega izmed modelov naučili na štirih petinah podatkov (metoda *clf.fit()*, in shranili napovedi na petini podatkov v spremenljivko *out_train*. V naslednji vrstici smo shranili še napoved na zunanjem testnem naboru v spremenljivko *proba_test*. Na ta način smo pridobili 5 različnih napovedi, od katerih smo izračunali povprečje.

```
    t_cv = list(StratifiedKFold(y_train, n_folds=5))
    for i, clf in enumerate(base_classifiers):
        cv_probabilities = np.zeros((X_test.shape[0]
        , len(t_cv)))
        # cross validation train
        for j, (train_i, test_i) in enumerate(t_cv):
            X_train_0 = X_train[train_i]
            y_train_0 = y_train[train_i]
            X_test_0 = X_train[test_i]
```

[12] https://en.wikipedia.org/wiki/Gradient_boosting

[13] https://github.com/log0/vertebral/blob/master/stacked_generalization.py

```
            # train each classifier
            clf.fit(X_train_0, y_train_0)
            # Get probabilities for click on internal data
            proba = clf.predict_proba(X_test_0)
            out_train[test_i, i] = proba[:, 1]
            # Probabilities for test data
            proba_test = clf.predict_proba(X_test)
            cv_probabilities[:, j] = proba_test[:, 1]
        # Average of predictions
        out_test[:, i] = cv_probabilities.mean(1)
```

V naslednjem koraku rezultate združimo z logistično regresijo. Uporabili smo stopnjo regularizacije 10, katero smo določili z poizkušanjem in preverjanjem točnosti. Z končnim združenim modelom smo napovedali verjetnosti razreda 1 na petini podatkov, ki smo jo v ta namen rezervirali na začetku. Sledil je še izračun točnosti AUC.

```
stack_clf = LogisticRegression(C=10)
stack_clf.fit(out_train, y_train)
stack_prediction = stack_clf.predict_proba(out_test)
```

Metoda skladanja je bila za samo izvajanje pričakovano najbolj procesorsko in časovno zahtevna, saj smo za vsakega izmed modelov na nivoju 0 zgradili 100 odločitvenih dreves, a je dala tudi najboljše rezultate.

4.4 Rezultati

Rezultati, ki smo jih dosegli pri napovedovanju so bili v skladu s začetnimi cilji, saj smo ciljali na AUC okoli 0,75. Na Zemantinem izzivu je naša metoda skladanja dosegla najboljši rezultat.

4.4.1 Merjenje točnosti napovedi

Za merjenje točnosti napovedi moramo najprej uvesti nekaj osnovnih mer[14]. Ko govorimo o točnosti modela imamo vedno v mislih razkorak med vrednostmi, ki jih model napove, in dejanske vrednosti. Poimenovanje teh mer prikazuje kontigenčna tabela na sliki 4.4.

[14] http://aimotion.blogspot.com/2010/08/tools-for-machine-learning-performance.html

	p' (Predicted)	n' (Predicted)
P (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Slika 4.4: Kontingenčna matrika.

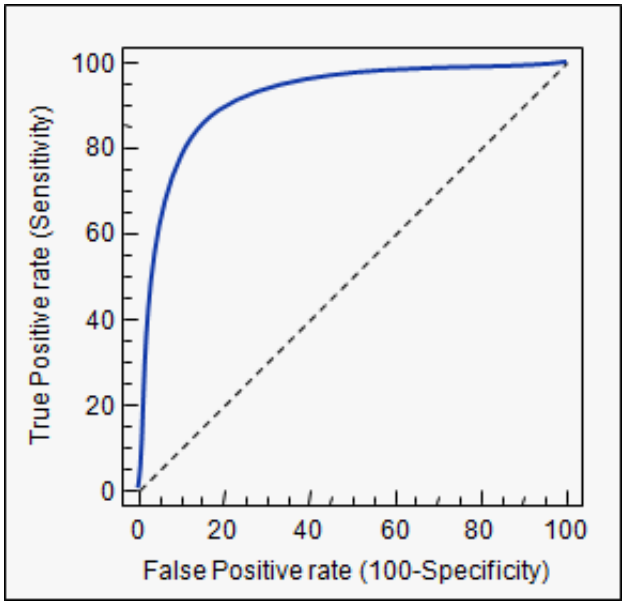
Za razumevanje mere, ki smo jo uporabljali, je pomemben delež zadetkov (TP) in delež napačno pozitivnih (FP). Omenjena deleža dobimo po enačbah 4.2 in 4.3.

$$TPR = TP/(TP + FN) \tag{4.2}$$

$$FPR = FP/(FP + TN) \tag{4.3}$$

Površina pod krivuljo ROC [5] (angl. *area under Receiving Operator Characteristics Curve*) je mera za ocenjevanje točnosti. Med drugo svetovno vojno so jo uporabljali za ocenjevanje dela operaterjev radarskih sistemov, ki so morali pravilno razlikovati med sovražnikovimi in domačimi letali. Kasneje se je uporabljala v medicini, v zadnjem času pa je zelo popularna v strojnem učenju in statistiki.

Krivulja ROC je določena kot razmerje med deležem zadetih 4.2 in deležem napačno pozitivnih 4.3. Na horizontalni osi beležimo FPR, na vertikalni pa TPR. Če vse primere razglasimo za pozitivne, smo dosegli FPR in TPR 1, če pa vse za negativne pa sta oba deleža 0. Za ostale vrednosti verjetja bosta FPR in TPR nekje med 0 in 1, zato moramo za vsak prag, pri katerem se FPR in TPR spremenita, deleža izračunati in ju vnesti v graf. Dobimo



Slika 4.5: Površina pod krivuljo ROC.

krivuljo iz slike[15] 4.5.

Površina pod to krivuljo torej določa točnost našega modela. Več napovedi, kot jih je model uvrstil med TPR, večja je površina pod krivuljo (angl. *Area under curve*) in posledično je naš model bolj natančen. Napovedni model, ki bi uvrščal popolnoma naključno, bi dosegel AUC 0.5, medtem ko bi model, ki bi vse primere uvrstil pravilno dosegel AUC 1.

4.4.2 Točnost napovednih modelov

Točnost modelov smo merili na 40 odstotkih učnih podatkov, preostanek pa je bil uporabljen za učenje. Vsak model smo testirali z 10 in 100 drevesi. Izkazalo se je, da povečanje števila dreves močno izboljša napovedno točnost. Rezultati so prikazani v tabeli 4.2.

Metoda naključnih gozdov je pričakovano dosegla slabše rezultate, saj smo pri skladanju poleg nje vključili še dve dodatni metodi. Je pa njena izvedba približno 3-krat hitrejša od metode skladanja pri enakem številu dreves, kar je

[15] `http://stats.stackexchange.com/questions/132777/`

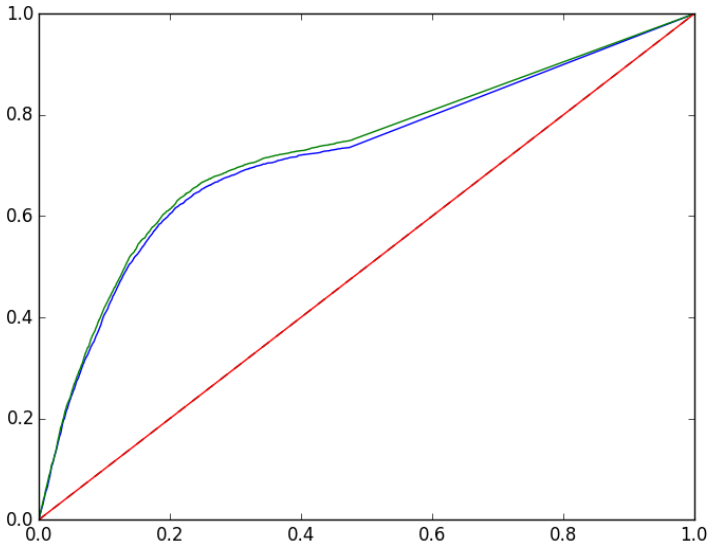
Tabela 4.2: Rezultati, ki so jih dosegli napovedni modeli.

Model	Število dreves	AUC
Naključni gozdovi	10	0.720
Naključni gozdovi	100	0.729
Skladanje	10	0.762
Skladanje	100	0.790

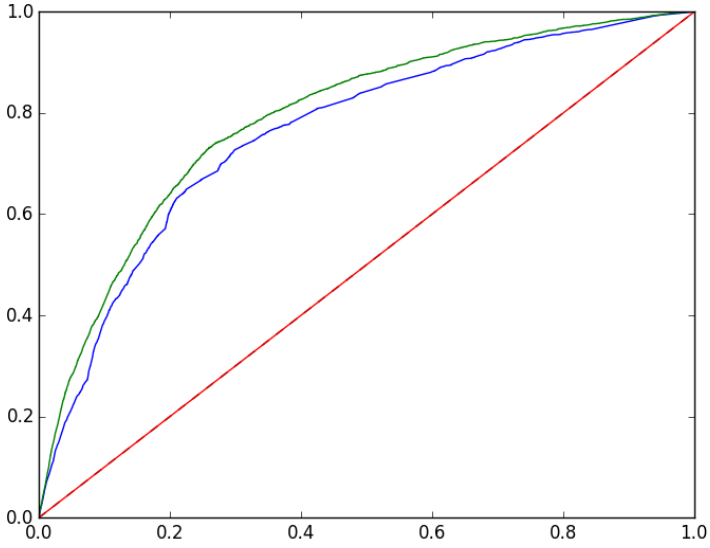
pričakovano, saj namesto enega pri skladanju učimo in napovedujemo z tremi algoritmi, na koncu pa uporabimo še logistično regresijo. Na sliki 4.6 vidimo krivuljo AUC za metodo naključnih gozdov z 10 drevesi (modra krivulja) in 100 drevesi (zelena krivulja). Opazimo, da imata modela do določenega praga precej enak rezultat, potem pa pride do manjše razlike, kjer več dreves da boljši rezultat.

Na sliki 4.7 je razvidno, da je imelo število dreves v posameznem klasifikatorju večji vpliv kot pri naključnih gozdovih.

Z skladanjem naključnih gozdov, Extra Trees in Gradient Boostinga smo dosegli najboljši rezultat na delu učnih podatkov in sicer 0.79, končni rezultat v okviru tekmovanja na skritih testnih podatki pa je bil 0.74. Slabši rezultat je najverjetneje posledica nekoliko specifičnih testnih podatkov.



Slika 4.6: Krivulja ROC za metodo naključnih gozdov z 10 in 100 drevesi.



Slika 4.7: Krivulja ROC za metodo skladanja za 10 in 100 dreves.

Poglavje 5

Sklepne ugotovitve

V diplomskem delu smo uspešno rešili zastavljen izziv z uporabo različnih tehnik podatkovnega rudarjenja in strojnega učenja. Sam problem se dejansko pojavlja v praksi. Zastavljen je bil s strani podjetja Zemanta, ki se ukvarja z razvojem platforme za dostavo oglasov in vsebin.

Najbolj specifične skupine, ki smo jih določili v prvem delu naloge, so zelo dobro pokazale, kako geografska lokacija vpliva na profil prebivalcev. Izkazalo se je, da je z kakovostnimi in dovolj raznolikimi vhodnimi podatki takšna segmentacija vsekakor možna. Vzpodbudno je tudi, da je uporaba skupin namesto poštnih številk močno izboljšala napovedno točnost.

Za informativno se je izkazala predstavitev rezultatov iskanja skupin na zemljevidu, ki je tudi potrdila, da so najdene skupine smiselne. Uspešno smo zaobšli omejitve prikaza velikega števila lokacij. Prikaz na zemljevidu bi bilo mogoče nadaljnjo izboljšati z uporabo Fusion tabel[1], s katerimi bi lahko območje vsake poštne številke obarvali, namesto uporabe žebljička.

Končni napovedni model (skladanje) je dosegel želeno napovedno točnost, čeprav so bili na testnih podatkih rezultati nekoliko slabši. Izboljšave bi bile možne predvsem z dodajanjem novih značilk, ki bi jih morda lahko pridelali iz atributa o podatku o strani, kjer se odpirajo številne možnosti.

Zaradi velike količine podatkov so nas vse skozi spremljale težave z zmo-

[1] `https://support.google.com/fusiontables`

gljivostjo računalnika, na katerem smo poganjali našo kodo. Glavni ozki grli sta bili procesorska moč in poraba pomnilnika, zato so bili potrebni nekateri kompromisi. Predvsem so dolgi časi izvajanja onemogočili natančnejše testiranje napovednih modelov in bolj natančno nastavljanje parametrov. Morda bi bilo v prihodnje smiselno razmisliti o najemu računskih virov v oblaku pri katerem od komercialnih ponudnikov, ki ponujajo tudi popuste za študente.

Na samem izzivu smo dosegli prvo mesto v svoji kategoriji in drugo mesto v končni razvrstitvi vseh treh izzivov, kar je dodatna potrditev našega dela.

Literatura

[1] Abbas Osama Abu. Comparison between data clustering algorithms. *The International Arab Journal of Information Technology*, 5(3):320–325, 2008.

[2] Frans Coenen. Data mining: Past, present and future. *The Knowledge Engineering Review*, 26(1):25–29, 2011.

[3] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[4] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques: concepts and techniques*. Elsevier, 2011.

[5] McNeil J. Barbara Hanley A., James. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Diagnostic Radiology*, 143(1):29–36, 1982.

[6] John A Hartigan and Manchek A Wong. Algorithm AS 136: A k-means clustering algorithm. *Applied Statistics*, pages 100–108, 1979.

[7] Douglas M Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004.

[8] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[9] Thorsten Joachims. Advances in kernel methods. chapter Making Large-scale Support Vector Machine Learning Practical, pages 169–184. MIT Press, Cambridge, MA, USA, 1999.

[10] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, volume 14, pages 1137–1145, 1995.

[11] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[12] Charles X Ling and Chenghui Li. Data mining for direct marketing: Problems and solutions. In *KDD*, volume 98, pages 73–79, 1998.

[13] Jiirg Sander Xiaowei Xu Martin Ester, Hans-Peter Kriegel. A density-based algorithm for discovering clusters. *KDD-96 Proceedings, AAAI*, 1996.

[14] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[15] Kai Ming Ting and Ian H. Witten. Stacked generalization: when does it work? In *in Procs. International Joint Conference on Artificial Intelligence*, pages 866–871. Morgan Kaufmann, 1997.

[16] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37–52, 1987.

[17] David H Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.